IIAI Open Conference Publication Series
IIAI Letters on Informatics and Interdisciplinary Research
Vol. 003, LIIR079
DOI: https://doi.org/10.52731/liir.v003.079

Effective Features for Finding Chord Interpretation Paths

Hiroyuki Yamamoto * † , Satoshi Tojo *

Abstract

We humans naturally feel tonality when we hear music, but its actual mechanism is still unknown. There have been many approaches to understand this mechanism, but one promising method is to measure the distance between chords, as was proposed in Tonal Pitch Space (TPS). Although the theory seems theoretically convincing, we could not know which features contribute most and which others are useless in an objective manner. In this study, we try to define a new distance model by clarifying the effective set of features among them. Based on the principle of the shortest distance, which is claimed by TPS, we estimate each distance by optimizing features to obtain the most plausible chord interpretation (i.e., a degree/key pair for a chord name). We propose the set of functions based on these features and we evaluate all simple combinations of them exhaustively. It turns out that accuracies almost max out with around 20 effective parameters in the functions, achieving about 80–90% accuracy, that outperforms the original TPS.

Keywords: Distance model, harmony analysis, tonal music.

1 Introduction

Can computers recognize music even though they lack auditory sense? Can they simulate human creativity and emotion out of music? Music is an attractive but hard target for the future of AI. In this paper, we consider obtaining *tonality* from a sequence of labelled chords; that is, to interpret each chord name by (*degree*, *key*) pair. We humans naturally feel tonality when we hear music. Then, our question is if tonality could be learned statistically by machines.

In order to find an adequate key, we consider how natural the connection between two chords are. Then, the numeric distance between two chords becomes a big hint. There have been a lot of approaches to applying some kinds of space to express harmonic features and utilizing the distance to calculate plausibility. Heinichen [1], Kellner [2], and Weber [3] tried to define the space to express the positional relationships of each key area (region). Riemann [4] applied the Tonnetz, which had been invented by Euler [5] as a way of representing just intonation, to analyze harmonic relationships from the viewpoint of pitch

* JAIST, Ishikawa, Japan
† yamamoto@kusuli.com

class (PC). The models of Bharucha and Krumhansl [6], and Deutsch [7] defines the distance between chords within the same region. Lerdahl [8] introduced Tonal Pitch Space (TPS) as another geometric model that considers the distances among degrees, regions, and constituent PCs.

A chord name can be interpreted in multiple ways, but we feel some interpretations are more preferable to others and the order of preference can be changed depending on the context (i.e., the relation with the other, presumably neighboring, chords). TPS claims the principle of the shortest distance, that is, we can find the most plausible chord interpretation as the one which minimizes the distances between neighboring chord interpretations. Utilizing this property of the distance defined by TPS, Sakamoto et al. [9] proposed a method to find the most plausible interpretation sequence for a chord sequence as the shortest path in the interpretation graph (Figure 1), that is a directed graph whose edges are weighted by the TPS' distances and expresses all possible chord interpretation sequences. However, it has been found that this method's prediction accuracy was only around 40%.

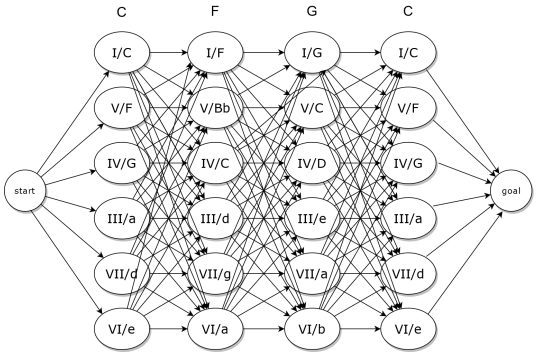

Figure 1: Interpretation graph.

Yamamoto and Tojo [10], assuming that one of the reasons is the calculation of TPS is based on classical music theory and/or expert's intuition but not on optimized with actual data, tried to find a better distance model by generalizing the distance function in TPS and proposed a method by which we can automatically fit its parameters with annotated dataset. They succeeded to find some combinations that achieve over 80% accuracy with less than a hundred parameters but, they tested only a few candidate combinations. In this study, we improve their approach by redefining the harmonic features and the way to combine them. This enables us to compare each harmonic feature and those combinations to find efficient use of features. We show some of them can achieve the same performance as the preceding work, with less number of effective parameters.

2 Preliminaries

2.1 Tonal Pitch Space

TPS is a music model for the quantitative harmony analysis proposed by Lerdahl [8]. It is proposed to complement his another music theory (the Generative Theory of Tonal Music [11]), which applies the generative grammar to extend the Schenkerian theory. A chord can be interpreted in multiple degree/key pairs (e.g., interpretations of C major triad are as

follows: I/C, III/a, V/F, IV/G, VI/e, and VII/d) and TPS defines a distance between every pair of these degree/key pairs[1]. TPS also claims the principle of the shortest distance which allows us to determine the most appropriate interpretation by finding the shortest distance.

The distance between chord interpretations x and y, when they are in related keys, can be calculated as equation 1.

$$\mathbf{tps}(x,y) = \mathbf{region}(x,y) + \mathbf{chord}(x,y) + \mathbf{basicspace}(x,y) \tag{1}$$

where $\mathbf{region}(x,y)$ is a distance between regions (i.e., keys with different scale notes), $\mathbf{chord}(x,y)$ is a distance between degrees, and $\mathbf{basicspace}(x,y)$ is a distance on a structure called basic space.

The calculation above is applicable only when x and y are in related keys which are defined as equation 2.

$$\mathbf{related\_keys}(R) = \begin{cases} \{\mathrm{I}, \mathrm{i}, \mathrm{ii}, \mathrm{iii}, \mathrm{IV}, \mathrm{V}, \mathrm{vi}\} & \text{if } R \text{ is a major key} \\ \{\mathrm{i}, \mathrm{I}, \flat\mathrm{III}, \mathrm{iv}, \mathrm{v}, \flat\mathrm{VI}, \flat\mathrm{VII}\} & \text{otherwise} \end{cases} \tag{2}$$

where roman numerals in this equation mean the keys with the tonic being the degree in key R (e.g., **related_keys**(F) is F, f, g, a, B♭, C, and d). If x and y are not in related keys (i.e., distant keys), the distance between x and y must be calculated as

$$\begin{aligned} \mathbf{tps}(x,y) &= \min\Big(\mathbf{tps}(x, T_{R_1}) + \Delta(R_1, R_n) + \mathbf{tps}(T_{R_n}, y) \\ &\qquad | R_1 \in \mathbf{related\_keys}(R_x), R_n \in \mathbf{related\_keys}(R_y) \Big) \\ \Delta(R_1, R_n) &= \min\Big(\sum_{i=1}^{n-1} \mathbf{tps}(T_{R_i}, T_{R_{i+1}}) | R_{i+1} \in \mathbf{related\_keys}(R_i) \Big) \end{aligned} \tag{3}$$

where T_R is R's tonic, R_z is chord interpretation z's key. In other words, the transition from x to y must be considered as a combination of transitions within related keys, and the overall distance is the shortest total distance of the transitions.

As explained above, the distance within related keys (Equation 1) is composed of the sum of three elements. Now, because Equation 3 is the sum of Equation 1s, the resulting distance can also be considered as the sum of three elements. Therefore, we can rewrite the distance as follows.

$$\mathbf{tps}(x,y) = \mathbf{tps\_region}(x,y) + \mathbf{tps\_chord}(x,y) + \mathbf{tps\_basicspace}(x,y) \tag{4}$$

2.2 Former Approaches based on TPS

Based on the distance defined by TPS and the principle of the shortest distance, Sakamoto et al. [9] have proposed a method to find the most plausible interpretation of a given chord sequence. Given a chord sequence, first, their method extends each chord to its interpretations and constructs a graph whose edges have weights that correspond to the distances on TPS. Then it applies the Viterbi algorithm to find the shortest interpretation paths from the start to the goal. Figure 1 shows an interpretation graph for chord sequence $\mathrm{C} \to \mathrm{F} \to \mathrm{G} \to \mathrm{C}$. One of the shortest interpretation paths in Figure 1 is $\mathrm{I/C} \to \mathrm{IV/C} \to \mathrm{V/C} \to \mathrm{I/C}$.

Catteau et al. [12] utilized the key profiles of Temperley [13] alongside TPS to define probabilities concerning chords, scales, and chroma vectors to estimate keys and chords

[1] We call degree/key pairs "chord interpretations" in this paper.

from audio. Rocher et al. [14] used Temperly's key profiles and TPS to construct a harmonic graph and then estimate individual chords and keys by finding the best path. In the effort to improve cadence detection, Matsubara et al. [15] have proposed to restrict the minor scale to harmonic one to avoid ambiguity in chord interpretations and to revise the candidates of interpretation of each chord.

Yamamoto and Tojo [10] have tried to generalize TPS and proposed several functions called "distance elements (DEs)" and a way to train them with annotated datasets. Based on the method of [9], their method replaces the TPS with the proposed generalized TPS then convert path distance to path probability such that the shortest path should have the highest probability, and finally apply SGD to update parameters. Their best model (i.e., a DE or combination of DEs) achieved over 86% accuracy while the original TPS was about 40%, and they also found a model with just 58 learnable parameters could achieve more than 80%. But they defined only 15 DEs (including those of the original TPS') and some combinations of them. Although those DEs are selected alongside the basic intuition of TPS, there are many more possible combinations to search for.

3 Our Approach

In this section, we describe the structure of the distance functions and their relation to the previous approach. First, we define some basic features of the transitions of chord interpretations (Section 3.1). Next, we define basic distance functions which classify a chord interpretation pair by using some of the basic features and return a distance value based on the classification (Section 3.2). Then we define the ways to combine distance functions (Section 3.3). Finally, we briefly explain the way how to embed the distance functions to the previous approach (Section 3.4).

3.1 Notions for Basic Features

Let $\mathscr{I}$ be a set of chord interpretations, each of which is represented by an ordered pair (Cartesian product) of major/minor, pitch class (PC) of tonic, and degree; where

$$\mathscr{M} = \{\text{minor}, \text{major}\}, \mathscr{T} = \{\text{C}, \text{C}\sharp, \cdots, \text{B}\}, \mathscr{D} = \{\text{I}, \text{II}, \cdots, \text{VII}\}. \tag{5}$$

Therefore, $\mathscr{I} = \mathscr{M} \times \mathscr{T} \times \mathscr{D} = \{(\text{minor}, \text{C}, \text{I}), (\text{minor}, \text{C}, \text{II}), \cdots, (\text{major}, \text{B}, \text{VII})\}$. First, we prepare a function to retrieve a component of each triple; **M** simply extracts the first argument of $\mathscr{I}$ and replaces minor and major with 0 and 1 respectively, **T** extracts the second argument and replaces $\text{C}, \text{C}\sharp, \cdots, \text{B}$ with $0, 1, \cdots, 11$, and **D** extracts the third argument and replaces roman $\text{I}, \text{II}, \cdots, \text{VII}$ for Arabic ones ignoring the case (i.e., upper or lower) then subtract 1:

$$\mathbf{M} : \mathscr{I} \to \{0, 1\}, \mathbf{T} : \mathscr{I} \to \{0, 1, \cdots, 11\}, \mathbf{D} : \mathscr{I} \to \{0, 1, \cdots, 6\}. \tag{6}$$

We also use Roman numeral notation such as "I/C" to express the elements of $\mathscr{I}$ for its brevity. For example, iii/C to express (major, C, iii), and VI/d, (minor, D, VI).

3.2 Basic Functions

Distance functions receive two chord interpretations (namely, source and destination of the transition) and return a real value for the transition distance (i.e., $\mathscr{I} \times \mathscr{I} \to \mathbb{R}$). These

functions are divided into two types, learnable and fixed. A learnable function is composed of a classifier and a set of learnable parameters. The function distinguishes a certain number of cases with the classifier and returns a value stored in the corresponding parameter. Therefore, if a learnable basic function distinguishes n cases, the function uses n parameters (so we describe the number that each learnable function can distinguish by indicating the number of parameters). Each learnable function has it's own array v to store the parameters (i.e., $v[i], 0 \leq i \leq n-1$). A fixed function, on the other hand, returns predefined values and does not have learnable parameters. We can take other distance models like TPS as the basic functions (as long as they have the same domain and codomain) by dealing with them as fixed functions.

In this study, we define three groups of learnable functions and three fixed functions, which we call basic functions.

3.2.1 Basic Functions for Major/minor Features

As the learnable functions about major/minor features, we define four functions as follows.

(1) **m_dest** is a function that considers only the second (destination) chord interpretation. Thus,

$$\mathbf{m\_dest}(x,y) \stackrel{\Delta}{=} v[\mathbf{M}(y)] \tag{7}$$

where $x, y \in \mathscr{I}$ (we use these variables in the same meaning from now on). Because this function only distinguishes two cases, the number of parameters is two. The rest are the combinations of the values of $\mathbf{M}(x)$ and $\mathbf{M}(y)$. (2) **m_asym** distinguishes all combinations of $\mathbf{M}(x)$ and $\mathbf{M}(y)$, that is,

$$\mathbf{m\_asym}(x,y) \stackrel{\Delta}{=} v[\mathbf{M}(x) * 2 + \mathbf{M}(y)] \tag{8}$$

resulting in four parameters. (3) **m_sym1** considers both $\mathbf{M}(x)$ and $\mathbf{M}(y)$ ignoring the direction,

$$\mathbf{m\_sym1}(x,y) \stackrel{\Delta}{=} v[\mathbf{M}(x) + \mathbf{M}(y)] \tag{9}$$

resulting in three parameters. (4) **m_sym2** does the same as **m_sym1** except this function equates all the cases where $\mathbf{M}(x)$ and $\mathbf{M}(y)$ are the same,

$$\mathbf{m\_sym2}(x,y) \stackrel{\Delta}{=} v[|\mathbf{M}(x) - \mathbf{M}(y)|] \tag{10}$$

As a result, **m_sym2** has two parameters.

3.2.2 Basic Functions for Tonic Features

Regarding tonic features, we define four functions as follows.

The first two functions are stepwise distances[2] based on tonic PCs[3]. (1) **t_oneway_steps** is the stepwise one-way distance between source tonic PC and destination tonic PC (12 parameters)

$$\mathbf{t\_oneway\_steps}(x,y) = v[(\mathbf{T}(y) - \mathbf{T}(x)) \mod 12] \tag{11}$$

[2]This "distance" is not the values returned from the functions but just the indices of parameters.

[3]We do not use the source or destination tonic PCs by themselves because we assume, in contrast with major/minor or degree features, the absolute position is neutral to the interpretations (e.g., we do not distinguish I/A → I/C and I/F → I/A♭).

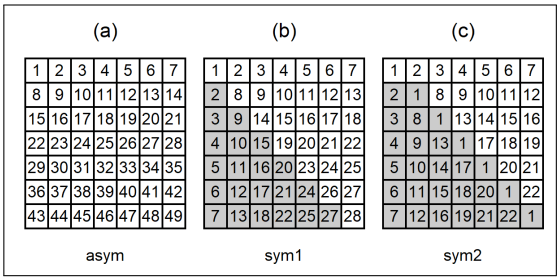

Figure 2: Combinations of $\mathbf{D}(x)$ (row) and $\mathbf{D}(y)$ (column). (a) Type **asym** distinguishes all the combinations. There are 49 independent parameters. (b) Type **sym1** ignores the direction. There are 28 independent parameters. (c) Type **sym2** ignores the direction and equates all the cases where $\mathbf{D}(x)$ and $\mathbf{D}(y)$ have the same values. There are 22 independent parameters.

And (2) **t_min_steps** ignores the direction (7 parameters)

$$\textbf{t\_min\_steps}(x,y) \stackrel{\Delta}{=} v\left[\min\begin{pmatrix}(\mathbf{T}(y)-\mathbf{T}(x)) & \mod 12,\\ 12-(\mathbf{T}(y)-\mathbf{T}(x)) & \mod 12\end{pmatrix}\right] \tag{12}$$

The other two functions are based on source and destination tonic PCs, but this time we use an auxiliary function

$$\mathbf{r}(x) \stackrel{\Delta}{=} \begin{cases}\mathbf{T}(x) & \text{if } \mathbf{M}(x)=1\\ (\mathbf{T}(x)+3) \mod 12 & \text{otherwise}\end{cases} \tag{13}$$

to distinguish regions. (3) **t_oneway_region_steps** is the stepwise one-way distance (12 parameters)

$$\textbf{t\_oneway\_region\_steps}(x,y) \stackrel{\Delta}{=} v[(\mathbf{r}(y)-\mathbf{r}(x)) \mod 12] \tag{14}$$

and (4) **t_min_region_steps** ignores the direction (7 parameters)

$$\textbf{t\_min\_region\_steps}(x,y) \stackrel{\Delta}{=} v\left[\min\begin{pmatrix}(\mathbf{r}(y)-\mathbf{r}(x)) & \mod 12,\\ 12-(\mathbf{r}(y)-\mathbf{r}(x)) & \mod 12\end{pmatrix}\right] \tag{15}$$

3.2.3 Basic Functions for Degree Features

About degree features, we define six functions as follows.

(1) **d_dest** distinguishes the degree value of the second chord interpretation (7 parameters)

$$\textbf{d\_dest} \stackrel{\Delta}{=} v[\mathbf{D}(y)] \tag{16}$$

The following five functions are the combinations of the features used above. (2) **d_asym** distinguishes all combinations of $\mathbf{D}(x)$ and $\mathbf{D}(y)$ as shown in Figure 2(a) (49 parameters). (3) **d_sym1** considers both $\mathbf{D}(x)$ and $\mathbf{D}(y)$ ignoring the direction as shown in Figure 2(b) (28 parameters). (4) **d_sym2** does the same as **d_sym1** except this function equates all the cases where $\mathbf{D}(x)$ and $\mathbf{D}(y)$ are the same as shown in Figure 2(c) (22 parameters).

Finally, we define stepwise distances of both degrees. (5) **d_oneway_steps** is based on the stepwise one-way distance between $\mathbf{D}(x)$ and $\mathbf{D}(y)$

$$\mathbf{d\_oneway\_steps}(x,y) \stackrel{\Delta}{=} v[(\mathbf{D}(y)-\mathbf{D}(x)) \mod 7] \tag{17}$$

(7 parameters). (6) **d_min_steps** does the same as **d_oneway_steps** except this function ignores the direction

$$\mathbf{d\_min\_steps}(x,y) \stackrel{\Delta}{=} v\left[\min\begin{pmatrix}(\mathbf{D}(y)-\mathbf{D}(x)) \mod 7,\\ 7-(\mathbf{D}(y)-\mathbf{D}(x)) \mod 7\end{pmatrix}\right] \tag{18}$$

(4 parameters).

3.2.4 Fixed Functions

We can use any functions as basic functions as long as they have the form of $\mathscr{I}^2 \to \mathbb{R}$, though they do not have learnable parameters (i.e., fixed). Here we define three fixed functions, **tps_region**, **tps_chord**, and **tps_basicspace** corresponding to the terms of Equation 4.

3.3 Combining Basic Functions

Basic functions defined above can be combined to create other functions (we call them compound functions, as distinguished from basic functions). Here, we define two operations, "addition" and "multiplication", to combine two functions. Both operations are commutative and associative, and "multiplication" has higher precedence than "addition".

By "addition", the resulting function just returns the sum of the two functions' results. So the number of necessary parameters is also the sum of those of the two functions. For example, if we combine **m_asym** and **t_min_steps** by "addition" (we denote this as "**m_asym** + **t_min_steps**"), we need $4+7=11$ parameters.

By "multiplication", the resulting function distinguishes all the combinations of the two functions. Therefore the number of necessary parameters is the product of those of the two functions. For example, if we combine **m_asym** and **t_min_steps** by "multiplication" (we denote this as "**m_asym** $\times$ **t_min_steps**"), we need $4 \times 7 = 28$ parameters. Note that we define this operation only on the learnable functions, so fixed functions can only be "added".

Because the set of functions is closed under these operations[4], these operations can be applied recursively. For example, **m_asym** $\times$ **t_min_steps** + **d_dest** is a valid function with $4 \times 7 + 7 = 35$ parameters. Likewise, **tps** (equation 4) can be constructed as a compound function **tps_region** + **tps_chord** + **tps_basicspace**. Additionally, functions like **m_asym** $\times$ **t_oneway_steps** $\times$ **d_asym** distinguish all possible cases in $\mathscr{I} \times \mathscr{I}$[5], so they subsume all other functions including those of TPS.

[4] In the case of learnable functions. As mentioned above, fixed functions can only be added.

[5] To be exact, there are $2 \times 12 \times 7 \times 2 \times 12 \times 7 = 28,224$ cases in total. But, as we mentioned above, we assume the absolute tonic position is neutral to the interpretation, it becomes only $2 \times 12 \times 7 \times 2 \times 7 = 2,352$ cases.

3.4 Path Probability and Training

With a ground truth interpretation path, we can construct an interpretation graph by restoring other candidate interpretations. And then we train the parameters to maximize the ground truth path's probability. As in [10], we convert the path cost to a path probability as follows

$$\mathbf{Pr}(P_{0:T} = p_{0:T} | G_{0:T}) \stackrel{\Delta}{=} \begin{cases} 1 & \text{if } T = 0 \\ \prod_{t=0}^{T-1} \frac{\exp(-\mathbf{Distance}(p_t, p_{t+1}))}{\sum_{l \in G_{t:t}} \sum_{m \in G_{t+1}} \mathbf{Pr}(P_t = l | G_{0:t}) \exp(-\mathbf{Distance}(l, m))} & \text{otherwise} \end{cases} \quad (19)$$

where **Distance** is a basic/compound function (this term is the only difference from [10]), T is the length of the interpretation path, G is the interpretation graph, P is a random variable (and we use lower case letters to denote observations) for the interpretation path (we can extract a part of G and P by indicating the start and end positions in the subscript, for example, $G_{a:b}$, and also $G_a = G_{a:a}$). Then we train the parameters by gradient descent on the cross entropy loss with the ground truth interpretation path.

Since our purpose is to find out the most plausible interpretation paths as the shortest paths in the interpretation graphs, any value can be added evenly to every cost value without changing the result. And this redundancy increases with each "addition". So we define effective parameters (EPs) by subtracting this redundancy from the number of parameters[6].

4 Experiments

4.1 Dataset

We use the dataset annotated in rntxt format [16], published at [17]. There are 384 pieces (1,905 phrases, 75,694 chords) and we regard every phrase as a unit (i.e., to which we predict the interpretation sequences) but when a phrase exceeds 50 chords we divide it into units each of which does not exceed 50 chords. Then we use 80% for training, 10% for validation, and the remaining 10% for the test.

We extracted key, degree, and applied chord information from rntxt. About applied chords, we expand them in three ways. In "original", every applied chord is converted into the prevailing key with the degree calculated by modulo operator. In "local", every applied chord is interpreted in the local key. And in "epsilon", a tonic chord is added at the end of every local key section to express pivot chord modulation or "ε-transition" proposed in [18]. After these expansions, we omit all repetitions of the same chord interpretations.

We set all initial parameter values to be zero and train them by mini-batch stochastic gradient descent with batch size=100 and learning rate=0.001. We continue training until no accuracy update in the validation set for an epoch[7] then pick the parameter which gives the highest validation accuracy.

4.2 Evaluation Targets

We evaluate all the simple combinations of three groups of basic functions exhaustively. The numbers of combinations are as follows: 175 for the zero or one term combinations[8],

[6] For example, $\mathbf{m\_asym} \times \mathbf{t\_min\_steps} + \mathbf{d\_dest}$ has 35 parameters and 2 redundancy, so the amount of effective parameters is $35 - 2 = 33$.

[7] We loosened the stopping condition because the original condition in [10] was too costly to conduct an exhaustive evaluation.

[8] $(4+1) \times (4+1) \times (6+1) = 175$.

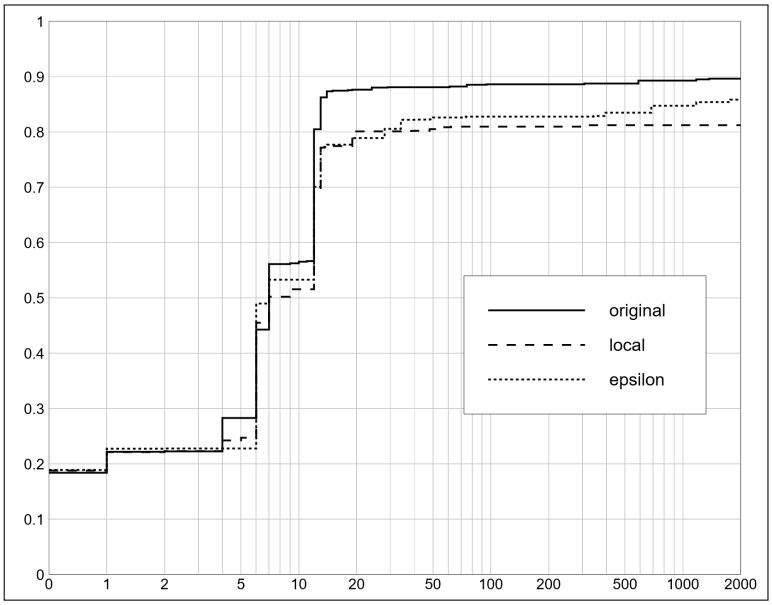

Figure 3: Best accuracy **y** with at most **x** EPs. Solid, dotted, and dashed lines represent the performances of "original", "local", and "epsilon" settings, respectively.

352 for two terms[9], and 96 for three terms[10]. In total, 623 combinations. And then we evaluate them all in three settings (i.e., "original", "local", and "epsilon").

4.3 Results and Discussion

Figure 3 shows the relationship between the number of EPs and the accuracy that could be achieved. Prediction in "original" setting seems to be the easiest. The accuracy passed 80% with only 12 EPs[11], and the best function[12] achieved 89.30%. The most difficult setting was "local" and the accuracy stopped increasing at 81.22%[13]. In "epsilon", though the chart moved almost as with that of "local" at first, it continued increasing and finally achieved 85.84%[14].

Next, we calculated how much each of the basic functions in Section 3.2 could increase the accuracy from nothing (**alone**) and on average (**average**) in "epsilon" setting (from now on, we mainly use the result from "epsilon" setting). Table 1(top) shows the result. Basically, they have higher scores on **average** than **alone**. Presumably, this is because basic functions have a kind of synergy effect when combined with basic functions of other features. Especially, **m_sym2** contributes only when it is combined with others. It is interesting that the functions which consider only one side (e.g., **d_dest**) had reasonable performances compared with the others. On the other hand, stepwise distances of degrees were almost useless, and the same goes for **tps_chord**. About the value of considering directions, we can compare directional groups (***_asym** and ***_oneway_***) and symmetric (i.e., non-directional) groups (***_sym*** and ***_min_**). Compared to their directional counterparts, symmetric func-

[9] $A+B$ type has $4\times4+4\times6+6\times4=64$, and $A\times B+C$ type has $(4\times4\times6)\times3=288$.
[10] $4\times4\times6=96$.
[11] with **t_min_region_steps** + **d_dest**.
[12] **m_asym** × **t_oneway_steps** × **d_asym** : 1,371 EPs. This is equivalent to DE8.2 proposed in [10].
[13] with **m_sym2** × **t_min_steps** × **d_sym2** : 307 EPs.
[14] with **m_sym1** × **t_oneway_region_steps** ×**d_asym** : 1,763 EPs.

Table 1: Performances of basic functions (+full TPS). **alone** is the accuracy gain (from 18.90% where all edges have 0 distance) when used alone. **average** is the average of, and **best** is the best accuracy gains (from the best accuracy without fixed functions, 85.94%) when combined with the other (learnable) functions.

basic function	params	alone	average	best
m_dest	2	3.84%	2.43%	
m_asym	4	3.88%	8.45%	
m_sym1	3	3.88%	8.38%	
m_sym2	2	0.00%	7.48%	
t_oneway_steps	12	12.63%	25.42%	
t_min_steps	7	12.34%	24.99%	
t_oneway_region_steps	12	20.46%	23.88%	
t_min_region_steps	7	20.39%	23.79%	
d_dest	7	20.84%	27.43%	
d_asym	49	25.60%	32.49%	
d_sym1	28	21.36%	31.26%	
d_sym2	22	21.35%	31.30%	
d_oneway_steps	7	0.51%	0.91%	
d_min_steps	4	-0.08%	1.13%	
tps_region		19.60%		-0.05%
tps_chord		0.44%		0.28%
tps_basicspace		19.70%		-0.43%
tps (equation 4)		19.74%		-0.24%

tions' average gains were a little lower while they can reduce a lot of EPs. Therefore, it is reasonable to use directional models when we predict, but it might be meaningful to use symmetric models to analyze trends in music because of their simplicity. Finally, about the functions from TPS (Table 1(bottom)), they, except **tps_chord**, worked well when used alone but almost failed to improve the best accuracy. This is because they can be subsumed by some cost functions as mentioned in Section 3.3.

As an example of learned parameters, we show that of $\mathbf{m\_sym2} \times \mathbf{t\_min\_steps} + \mathbf{d\_sym2}$, which was the smallest model that exceeded 80%, in Table 2. This function has $14 + 22 - 2 = 34$ EPs, and is symmetric. And Figure 4 shows two examples of predictions made by this model. (a) has a modulation and the function correctly predicted all but the exact moment when the modulation occurred. In (b), there is a secondary chord so we show the predictions in three settings. Prediction in "original" in this example only needs chord inter-

Table 2: Learned parameters of $\mathbf{m\_sym2} \times \mathbf{t\_min\_steps} + \mathbf{d\_sym2}$ (34 EPs). **c1**, **r1**, **c2**, and **r2** are $\mathbf{m\_sym2}(x,y)$, $\mathbf{t\_min\_steps}(x,y)$, $\mathbf{D}(y)$, and $\mathbf{D}(x)$ respectively. The values are shifted so that the minimum value is 0.

	(c1=)0	1		(c2=)0	1	2	3	4	5	6
(r1=)0	0.0000	3.4579	(r2=)0	2.8867	0.8288	1.7369	1.0132	0.0000	1.0121	0.2020
1	3.3310	3.9650	1	0.8288	2.8867	1.8827	1.4904	1.0528	1.6762	1.7790
2	4.2965	2.8998	2	1.7369	1.8827	2.8867	1.8570	1.6092	2.1316	1.9546
3	3.8512	2.9294	3	1.0132	1.4904	1.8570	2.8867	1.0696	1.5745	1.3540
4	3.5310	3.5929	4	0.0000	1.0528	1.6092	1.0696	2.8867	1.1286	1.1998
5	2.2645	3.2413	5	1.0121	1.6762	2.1316	1.5745	1.1286	2.8867	1.5999
6	3.2641	3.5508	6	0.2020	1.7790	1.9546	1.3540	1.1998	1.5999	2.8867

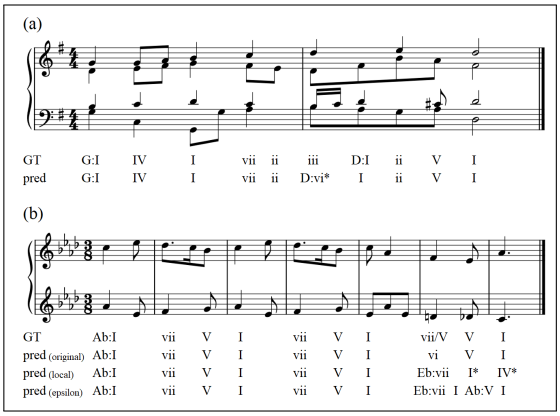

Figure 4: Examples of ground truth chord interpretations (GT) and the predictions (pred). Chord interpretations that the model failed to predict correctly are highlighted with "*".

pretations in A♭ key and the function succeeded to predict all of them. Prediction in "local" must have a different key at the secondary chord and the function predicted it correctly, but it failed to return to the original key after that. On the other hand, "epsilon" adds I/E♭ after the secondary chord, and the function predicted the whole interpretation successfully.

5 Conclusion

In this study, we have refined the distance functions in [10], which were to measure the chord distance proposed in the Tonal Pitch Space (TPS), in a systematic way. We defined three basic features of chord interpretation pairs, proposed three groups of basic functions, and then evaluated all possible combinations of them. It turned out that the new efficient distance models could achieve 80–90% accuracy which outperformed that of original TPS. Furthermore, we could construct them with only around 20 effective parameters (EPs). We also confirmed that the "ε-transition" indeed worked well.

This research concerns a big question, that is, if the computers can simulate our creativity and emotion by music. Toward this, we have formalized a statistical way to obtain tonality by machines. Although this is only the first step to the big question, we have shown a concrete and solid algorithm for a computer to guess the tonality properly. Our future work includes, (1) augmenting the structure of chord interpretations thereby it can be possible to express more detailed harmonic relationships, (2) considering the connections between chroma vectors and chord interpretations so that we can improve not only the expressiveness but also the potential range of application.

Acknowledgments

This research is supported by JSPS Kaken 21H03572 and 20H04302.

References

[1] J. D. Heinichen, *General-bass in der composition*. Dresden: J. D. Heinichen, 1728.

[2] D. Kellner, *Treulicher Unterricht im General-Bass*. Hamburg: C. Herold, 1737

[3] G. Weber, *Versuch einer geordneten theorie der tonsetzkunst*. Mainz: B. Schotts Söhne, 1821-24.

[4] H. Riemann, *Grosse kompositionslehre, Vol. 1*. Berlin: W. Spemann, 1902.

[5] L. Euler, *Tentamen novae theoriae musicae*. St. Petersburg Academy, 1739.

[6] J. Bharucha and C. Krumhansl, "The representation of harmonic structures in music: Hierarchies of stability as a functions of context", *Cognition*, 13(1), pp.63-103, 1983.

[7] D. Deutsch, "The processing of pitch combinations", in *The Psychology of Music*, Chapter 10, Academic Press, New York, 1999.

[8] F. Lerdahl, *Tonal pitch space*. New York, Oxford University Press, 2001.

[9] S. Sakamoto, S. Arn, M. Matsubara, S. Tojo, "Harmonic analysis based on tonal pitch space", in *Proc. of the 8th International Conference on Knowledge and Systems Engineering (KSE)*, pp.230-233, 2016.

[10] H. Yamamoto, S. Tojo, "Generalized tonal pitch space with empirical training", in *Proc. of the 18th Sound and Music Computing Conference (SMC)*, pp.300-307, 2021.

[11] F. Lerdahl, R. Jackendoff, *A Generative Theory of tonal music*. Cambridge, MA, 1983.

[12] B. Catteau, J. Martens, M. Leman, "A probabilistic framework for audio-based tonal key and chord recognition", in *Advancesin Data Analysis*, pp.637-644, 2007.

[13] D. Temperley, "What ' s Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered", in *Music Perception* 17, 1, pp.65-100, 1999.

[14] T. Rocher, M. Robine, P. Hanna, L. Oudre, "Concurrent estimation of chords and keys from audio", in *Proc. of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, pp.141-146, 2010.

[15] M. Matsubara, T. Kodama, S. Tojo, "Revisiting cadential retention in GTTM", in *Proc. of the 8th International Conference on Knowledge and Systems Engineering (KSE)*, pp.218-223, 2016.

[16] D. Tymoczko, M. Gotham, M. S. Cuthbert, C. Ariza, "The romantext format: a flexible and standard method for representing Roman numeral analyses", in *Proc. of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, pp.123-129, 2019.

[17] M. Gotham, R. Kleinertz, C. Weiss, M. Müller, S. Klauk. "What if the 'When' implies the 'What'?: human harmonic analysis datasets clarify the relative role of the separate steps in automatic tonal analysis", in *Proc. of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, pp.229–236, 2021

[18] H. Yamamoto, Y. Uehara, S. Tojo, "Jazz harmony analysis with ε-transition and cadential shortcut", in *Proc. of the 17th Sound and Music Computing Conference (SMC)*, pp.316-322, 2020.